IN THE CLAIMS:

Please cancel claims 2-3, 10 and 21-25 and amend the claims as follows:

1. (Currently Amended) A method of processing graphics data in a graphics processing unit including a fragment processing unit associated with a conflict detection unit and ~~using~~ a buffer comprising:

receiving ~~fragments,~~ a fragment associated with a location in the buffer;

tracking in the conflict detection unit a pending write to the location in the buffer;

shading at least a portion of the fragments to produce shaded fragment data;

waiting to read the location in the buffer until the pending write to the location in the buffer is completed;

updating the conflict detection unit when the pending write to the location is completed;

shading the fragment using data read from the location in the buffer to produce additional shaded fragment data;

writing the shaded fragment data to at least one location in the buffer; and

writing the additional shaded fragment data to a location in the buffer.

2 - 3 (Cancelled)

4. (Currently Amended). The method of claim 1, wherein data stored in the location in the buffer is also stored in an entry in a data cache, the data cache storing locations to which writes are pending, the locations being accessible to the fragment processor.

5. (Original) The method of claim 4, further comprising:

invalidating the entry in the data cache associated with the pending write to the location in the buffer.

6. (Original) The method of claim 4, further comprising:

updating the entry in the data cache when the pending write to the location is completed.

7. (Currently Amended) A method for processing fragments under control of a fragment program in a fragment processing unit, comprising:

processing a first fragment as specified by the fragment program;

determining in a conflict detection unit that a write to a location in a buffer utilized in the processing is pending prior to reading the location in the buffer;

storing an instruction for processing the first fragment waiting for the write to complete;

processing another fragment as specified by the fragment program while waiting for the write to complete;

reading, responsive to the conflict detection unit, the location in the buffer; and

processing responsive to the conflict detection unit, the first ~~a~~ fragment in the fragment processing unit as specified by the fragment program upon determining that the write to the buffer location is complete.

8. (Currently Amended) The method of claim 7, wherein the fragment program performs depth buffering prior to shading, and the processing of a fragment by the fragment processing unit is carried out only if a depth value of the fragment is closer to the viewpoint than an established value comprising establishing a depth buffer value at a position.

9. (Original) The method of claim 8, wherein the fragment program performs depth peeling.

10. (Cancelled)

11. (Original) The method of claim 8, wherein the buffer is one of several buffers stored in graphics memory.

12. (Original) A programmable graphics processor for execution of program instructions comprising:

a conflict detection unit configured to selectively store at least a portion of a position associated with a plurality of fragments and generate a position conflict status for each of the plurality of fragments;

a read interface responsive to the positions stored by the conflict detection unit and configured to read data associated with one of the positions from a graphics memory and output the data to a fragment processing unit;

the fragment processing unit configured to receive a fragment associated with the one position, and the data from the read interface and generate a processed fragment; and

a write interface configured to write the processed fragment to the graphics memory.

13. (Original) The programmable graphics processor of claim 12, wherein the portion of a position specifies a region of fragment positions.

14. (Original) The programmable graphics processor of claim 12, wherein the read interface is configured to read data responsive to the position conflict status.

15. (Original) The programmable graphics processor of claim 12, wherein a position stored in the conflict detection unit includes at least a buffer identifier and a pair of coordinates.

16. (Currently Amended) The programmable graphics processor of claim 12, wherein the fragment processing unit further includes a data cache configured to store data entries, each data entry associated with a position in a buffer to track pending writes to the buffer, the position being accessible to the fragment processing unit.

17. (Original) The programmable graphics processor of claim 16, wherein the data cache is configured to invalidate a data entry associated with a position in a buffer when a write is pending for the position in the buffer, producing an invalid data entry.

18. (Original) The programmable graphics processor of claim 16, wherein the data cache is configured to read data from the position in the buffer and store the data read in the invalid data entry associated with the position in the buffer.

19. (Original) The programmable graphics processor of claim 16, wherein the data cache is configured to update the entry in the data cache when the write to the position in the buffer is completed.

20. (Original) The programmable graphics processor of claim 15, wherein the conflict detection unit includes a hash unit.

21-25 (Cancelled)

26. (New) A method as claimed in claim 1, wherein each fragment comprises pixels and is associated with a tile, a display comprising a plurality of tiles, further comprising

associating through the conflict detection unit a coverage mask with each tile, the mask representing pending units to each of the pixels within the tile,

storing coverage match data for each of a plurality of tiles in the display.

27. (New) A method as in claim 26 further comprising transferring tile data entries and the coverage mask data for the tile to the fragment processor and claiming the stored tile data and coverage mask data after the data is transferred for display.

28. (New) A method as claimed in claim 27 further comprising assigning a issue timestamp and a retire timestamp to each tile to establish a timestamp window for the tile within which data for the tile is received.

29. (New) A method as claimed in claim 1 further comprising the step of executing a pixel load instruction to configure the conflict detection unit to detect a position conflict for a position, the pixel load instruction including a second destination to address for storing the data until the position conflict is resolved.

30. (New) A programmable graphics processor as claimed in claim 12 further comprising resources controlled by the conflict detection unit configured to store a fragment program instruction that depends for execution on data having a position conflict identified by the conflict detection unit.

31. (New) A programmable graphics processor as claimed in claim 12 further comprising resources controlled by the conflict detection unit to store a fragment associated with a position for which a position conflict exists.

32. (New) A method as claimed in claim 1 wherein the conflict detection unit is configured to cause the fragment processor to perform a depth test and a stencil test on the fragment prior to writing the data to the buffer location, and to terminate the write if the fragment fails the depth test or stencil test.